CS 3430: Py & PL
Assignment 10: Generating Octave Plots

Vladimir Kulyukin
Department of Computer Science
Utah State University

1. Learning Objectives

1. Regular Expressions
2. Pipelines
3. Plotting Data

2. Generation of Data Plots

I originally planned to dedicate this assignment to modules & packages but changed my mind after talking to a few students about their final projects. Many of them asked me about Python/Perl plotting tools. If you really want to use Py/PL for plotting, then, for Python, you may want to check https://plot.ly/python/ and, for PL, go to www.cpan.org and search to your heart's content until you find something you like. I would like, however, to show you an alternative, which is to write Py and PL programs to generate plotting scripts for other plotting tools and use those tools to do the graphing for you. It will also give those of you who struggled with regexes and pipelines in the last two assignments and the 2nd coding exam more practice and a chance to earn more points.

The plotting tool that we will use in this assignment is Octave, an open source equivalent of Matlab. Go to http://www.gnu.org/software/octave/download.html and spend five minutes to install this great free software on your computer. Once you have it installed, start it and do **File | New Script** to create a new script and call it **histo_01.m**. All Octave script files have the extension **.m** (this stands for module).

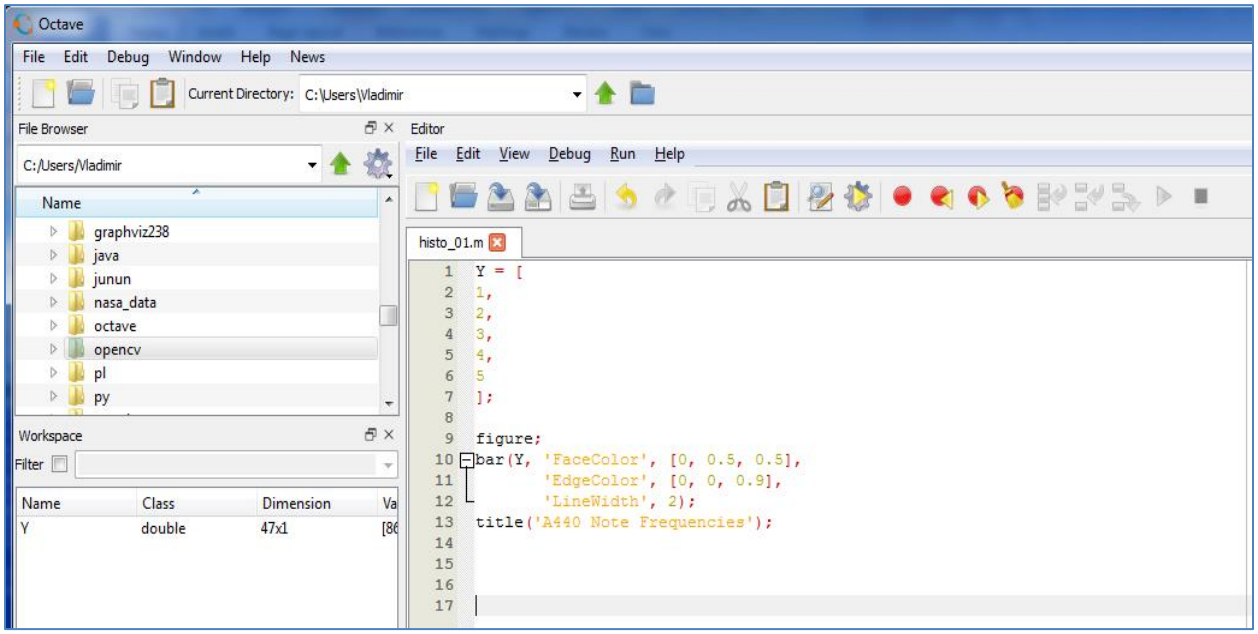

Figure 1. Octave code editor

Then in the editor window, shown in **Figure 1**, type (or copy and paste) the following code and save it.

```
Y = [
1,
2,
3,
4,
5
];

figure;
bar(Y, 'FaceColor', [0, 0.5, 0.5],
    'EdgeColor', [0, 0, 0.9],
    'LineWidth', 2);
title('A440 Note Frequencies');
```

This code is an Octave script that draws a histogram. If you have never heard the term histogram, do not worry. It is a pretty intuitive concept you can quickly familiarize yourself with by reading this great Wiki article - https://en.wikipedia.org/wiki/Histogram. The array Y in the above script defines the values of the histogram's bars. Specifically, the first bar has a height of 1, the 2nd bar has a height of 2, etc. The lines beginning with figure and ending with title define a figure that draw the histogram with a specific face color, edge color, and line width. When you click on the run error and run the script, you should see the plot shown in **Figure 2**.

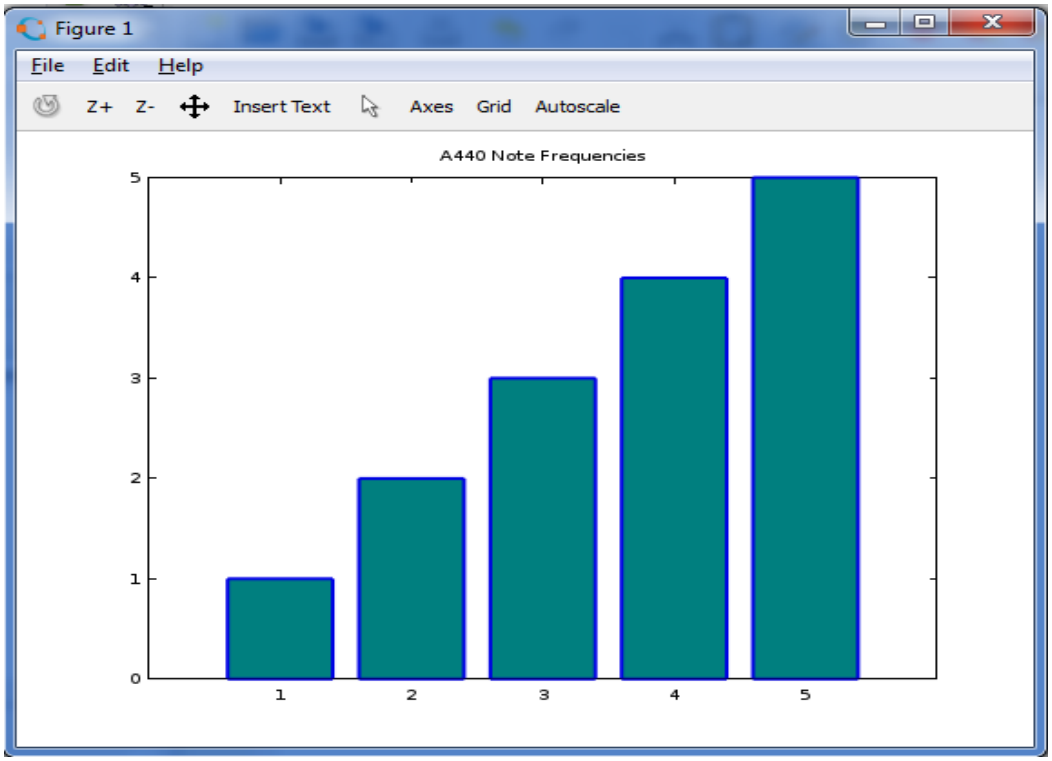

Figure 2. Histogram plot generated by histo_01.m

2. Automating Generation of Data Plots with Python and Perl

We will implement small Py and PL pipelines that generates Octave histogram scripts. For the data files, let us go back to Assignment 7 where we did some computer music analysis. In particular, we will generate octave scripts from text files that look as follows:

```
2015-08-15_15-41-32 44100 B2 C3# D3 D3#
2015-08-15_15-41-32 44100 A1 A1# B1 D1# E1 F1 F1# G1 A2 B2 F2# G2 C3 C3# D3 D3#
2015-08-15_15-41-32 44100 C1# D1 D1# B2 C3# D3 D3#
2015-08-15_15-41-32 44100 C3# D3 D3#
2015-08-15_16-01-32 44100 C3 C3# D3
2015-08-15_16-01-32 44100 B2 C3 C3# D3
2015-08-15_16-01-32 44100 C3 C3# D3
2015-08-15_16-01-32 44100 C3 C3# D3
2015-08-15_15-41-32 44100 B2 C3# D3 D3#
2015-08-15_15-41-32 44100 A1 A1# B1 D1# E1 F1 F1# G1 A2 B2 F2# G2 C3 C3# D3 D3#
2015-08-15_15-41-32 44100 C1# D1 D1# B2 C3# D3 D3#
2015-08-15_15-41-32 44100 C3# D3 D3#
2015-08-15_15-41-32 44100 C3# D3
```

The first element on each line is a time record, the second is frequency, i.e., 44100, the other elements are the notes. I have included three such files with this assignment: timestamped_notes_11025.txt, timestamped_notes_44100.txt, and timestamped_notes_5512.txt.

2.1 FLTR_TIME_NOTE_RECS.PL & FLTR_TIME_NOTE_RECS.PY

Reuse as much of your code from Assignment 7 to write the Py and PL scripts **fltr_time_note_recs.py** and **fltr_time_note_recs.pl** that transforms timestamped notes files as follows:

```
>fltr_time_note_recs.pl timestamped_notes_11025.txt
2015 08 15 15 41 32 11025 A0# B0 C1 C1# D1 D1# E1
2015 08 15 15 41 32 11025 A0 A0# B0 C1 C1# D1 D1# E1 C2 C2#
2015 08 15 15 41 32 11025 A0 A0# B0 C1 C1# D1 D1#
2015 08 15 15 41 32 11025 A0 A0# B0 C1 C1# D1 D1# E1 G1#
2015 08 15 15 41 32 11025 A0# B0 C1 C1# D1 D1# E1 C2# D2
2015 08 15 15 41 32 11025 A0 A0# B0 C1 C1# D1 D1# E1 F1 F1# G1# C2 D2 D2# E2 F2# G2
2015 08 15 16 01 32 11025 A0 A0# B0 C1 C1# D1
2015 08 15 16 01 32 11025 A0# B0 C1 C1# D1 D1# E1 C2
2015 08 15 16 01 32 11025 A0 A0# B0 C1 C1# D1 D1#
2015 08 15 16 01 32 11025 A0 A0# B0 A1# B1 C1 C1# D1 D1# E1

>python fltr_time_note_recs.py < timestamped_notes_11025.txt
2015 08 15 15 41 41 11025 32 11025 A0# B0 C1 C1# D1 D1# E1
2015 08 15 15 41 41 11025 32 11025 A0 A0# B0 C1 C1# D1 D1# E1 C2 C2#
2015 08 15 15 41 41 11025 32 11025 A0 A0# B0 C1 C1# D1 D1#
2015 08 15 15 41 41 11025 32 11025 A0 A0# B0 C1 C1# D1 D1# E1 G1#
2015 08 15 15 41 41 11025 32 11025 A0# B0 C1 C1# D1 D1# E1 C2# D2
```

```
2015  08  15  15  41  41  11025  32  11025  A0 A0# B0 C1 C1# D1 D1# E1 F1 F1# G1# C2 D2 D2# E2 F2# G2
2015  08  15  16  01  01  11025  32  11025  A0 A0# B0 C1 C1# D1
2015  08  15  16  01  01  11025  32  11025  A0# B0 C1 C1# D1 D1# E1 C2
2015  08  15  16  01  01  11025  32  11025  A0 A0# B0 C1 C1# D1 D1#
2015  08  15  16  01  01  11025  32  11025  A0 A0# B0 A1# B1 C1 C1# D1 D1# E1
```

2.2 COMPUTE_NUMBERED_NOTE_FREQ_TOOPS.PY & COMPUTE_NUMBERED_NOTE_FREQ_TOOPS.PL

Now write scripts **compute_numbered_note_freq_toops.py** and **compute_numbered_note_freq_toops.pl** that takes the output of **fltr_time_note_recs** and outputs tab-separated toops where the first element is the number of a note on the 88 piano keyboard and the second number is the frequency of occurrence of that note in the tab-separated records output by **fltr_time_note_recs**. You can use the Py dictionary and PL hash called A440_NOTE_NUMBERS in **compute_numbered_note_freq_toops.py** and **compute_numbered_note_freq_toops.pl** to convert each note to its number on the 88 piano keyboard.

```
>fltr_time_note_recs.pl timestamped_notes_11025.txt | compute_numbered_note_freq_toops.pl
19    1
18    2
22    1
5     10
6     10
10    1
17    2
2     10
8     7
20    1
15    1
16    3
1     7
23    1
12    2
7     9
14    1
9     1
3     10
4     10
```

Here is the Py output:

```
>python fltr_time_note_recs.py < timestamped_notes_11025.txt | python compute_numbered_note_freq_toops.py
1     7
2     10
3     10
4     10
5     10
6     10
7     9
```

8 7
9 1
10 1
12 2
14 1
15 1
16 3
17 2
18 2
19 1
20 1
22 1
23 1

The above output simply means that note whose number is 1 on the 88 keyboard, e.g. A0, has a frequency of occurrence equal to 7, etc.

2.3 SORT_NUMBERED_NOTE_FREQ_TOOPS.PY & SORT_NUMBERED_NOTE_FREQ_TOOPS.PL

Now write scripts **compute_numbered_note_freq_toops.py** and **compute_numbered_note_freq_toops.pl** that sorts the note numbers in ascending order. Here is the Py output. The PL output is the same.

>python fltr_time_note_recs.py < timestamped_notes_11025.txt | python compute_numbered_note_freq_toops.py | python sort_numbered_note_freq_toops.py
1 7
2 10
3 10
4 10
5 10
6 10
7 9
8 7
9 1
10 1
12 2
14 1
15 1
16 3
17 2
18 2
19 1
20 1
22 1
23 1

2.4 GEN_OCTAVE_HISTO.PY & GEN_OCTAVE_HISTO.PL

The final cut is to generate Octave histogram plotting scripts from the tab-separated toops output by **sort_numbered_note_freq_toops**. The script takes a file path where it stores the generated Octave script. Here is the PL pipeline in DOS:

```
>fltr_time_note_recs.pl timestamped_notes_11025.txt |  compute_numbered_note_freq_toops.pl |
sort_numbered_note_freq_toops.pl | gen_octave_histo.pl ..\..\octave\pl_histo.m
```

Here is the Py pipline in DOS:

```
>python fltr_time_note_recs.py < timestamped_notes_11025.txt | python compute_numbered_note_freq_toops.py |
python sort_numbered_note_freq_toops.py | python gen_octave_histo.py ..\..\octave\py_histo_11025.m
```

Both of these files generate the files **pl_histo_11025.m** and **py_histo_11025.m**. They are identical and look as follows:

```
Y = [
        7,
        10,
        10,
        10,
        10,
        10,
        9,
        7,
        1,
        1,
        2,
        1,
        1,
        3,
        2,
        2,
        1,
        1,
        1,
        1,
];
figure;
bar(Y, 'FaceColor', [0, 0.5, 0.5],
'EdgeColor', [0, 0.5, 0.5],
'LineWidth', 2);
title('A440 Note Frequencies');
```

When you graph these files in Octave, you will see the following histogram plots as shown in **Figure 3**.

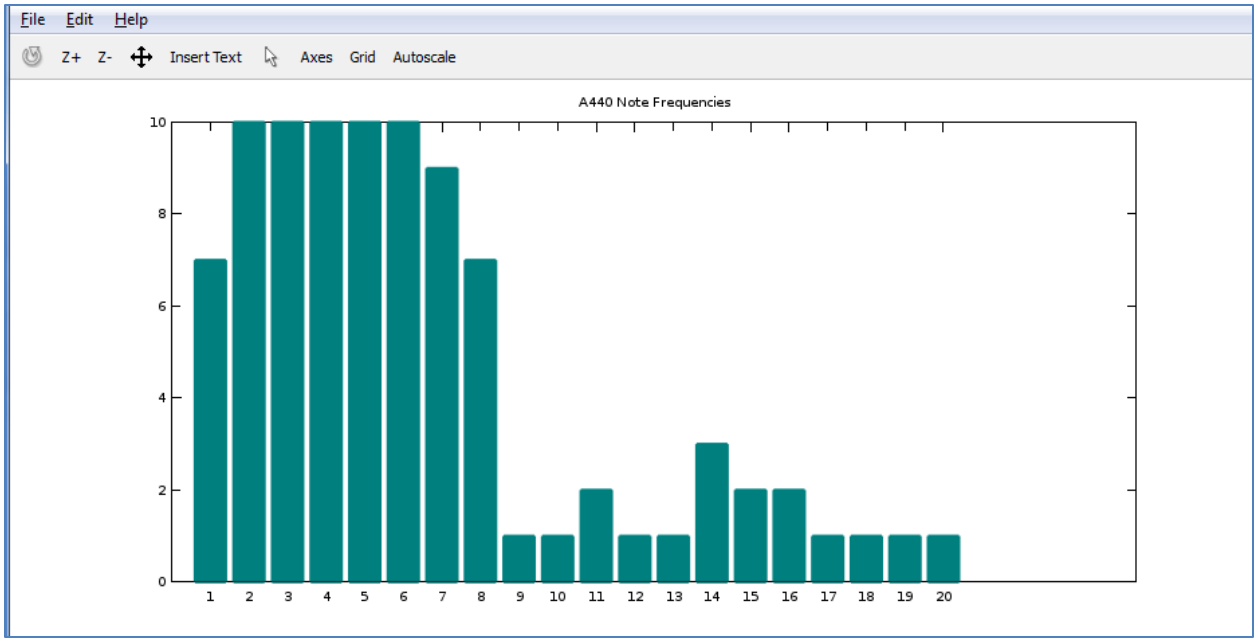

Figure 3. Histogram plot generated from py_plot.m and pl_plot.m

3. What and Where to Submit

Submit through Canvas **fltr_time_note_recs.py/pl**, **compute_numbered_note_freq_toops.py/pl**, **sort_numbered_note_freq_toops.py/pl**, and **gen_octave_histo.py/pl** and your six Octave scripts, two for each timestamped file, i.e., **py/pl_histo_5512.m, py/pl_histo_11025.m**, and **py/pl_histo_44100.m**.